ＢＣＲＸ６３０＿１４４　マイコン開発セット　マニュアル

第１版２０１４．３．１　　　　**第１版**

【　製品概要　】

本マニュアルはＢＣＲＸ６３０＿１４４　ＣＰＵボードのソフトウエア開発を行うために必要なソフトウエアインストゥール手順、添付ＣＤのサンプルプログラムの動作について解説されています。サンプルプログラムはルネサスエレクトロニクス社が無償で提供するＨＥＷ４＋Ｃコンパイラを使用します。本ＣＰＵボード開発にはルネサスエレクトロニクス社製Ｅ１が必要です。



１．開発環境、事前準備

１－１．　開発環境

ａ：開発セット　同梱物
ｂ：ＢＣＲＸ６３０＿１４４　ＣＰＵボードの特徴
ｃ：デバッカＥ１
ｄ：無償のＨＥＷ、ＲＸ用Ｃコンパイラのダウンロード
ｅ：ＣＤコピー、デバイスドライバＤ２ＸＸのインストゥール

１－２　動作、デバック

ａ：ＨＥＷ起動、コンパイル、書き込み、動作
ｂ：ブレークポイント設定、レジスタ、変数参照概要
ｃ：新しいプログラムを作る

２．サンプルプログラム

２－１．　ｓａｍｐｌｅ１　出力ポートのＯＮ，ＯＦＦ
２－２．　ｓａｍｐｌｅ２　ＳＩＯ（ＵＳＢ）でパソコンとやりとり
　２－２－１　ｓａｍｐｌｅ２１　ＳＩＯ（ＲＳ２３２Ｃ）でパソコンとやりとり
　２－２－２　ｓａｍｐｌｅ２２　ＥＥＰＲＯＭ（２５ＬＣ２５６）読み書き
２－３．　ｓａｍｐｌｅ３　Ａ／Ｄ変換をＵＳＢ出力、ＲＳ２３２Ｃ出力
２－４．　ｓａｍｐｌｅ４　割り込み
２－５．　ｓａｍｐｌｅ５　ＰＷＭ出力
２－６．　ｓａｍｐｌｅ６　三角、対数、平方根関数を使う
２－７．　ｓａｍｐｌｅ７　Ｄ／Ａにｓｉｎ，ｃｏｓ演算した正弦波を出力する

１－１．開発環境

ａ：開発セット同梱物

ＲＸ６３０＿１４４　ＣＰＵボード
ＣＤ（サンプルプログラム、デバイスドライバ、ドキュメント）
マニュアル（本誌）
ハードウエアマニュアル
電源ケーブル
Ｋケーブル（ＲＳ２３２Ｃ用）

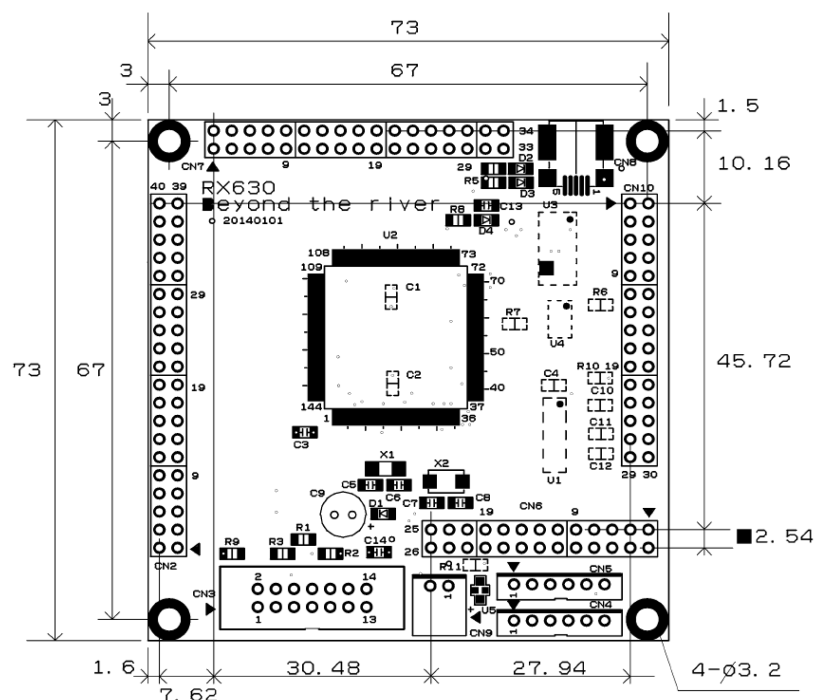


※開発に必要なルネサスエレクトロニクス社製デバッカＥ１は同封されておりません。別途必要です。

ｂ：ＢＣＲＸ６３０＿１４４　ＣＰＵボードの特徴

●ＲＸアーキテクチャコア（３２ビットシングルチップＣＩＳＣ　最大１６５ＤＭＩＰＳ　１００ＭＨｚ動作時）、１４４ピン、Ｒ５Ｆ５６３０ＤＤＤＦＢ搭載。
●３２ビット単精度浮動小数点（ＩＥＥＥ７５４準拠）、２種類の積和演算器（メモリ間、レジスタ間）、３２ビット乗算器（最速１クロックで実行）、５段パイプラインのＣＩＳＣハーバードアーキテクチャ
●外部クリスタルメイン１２．５ＭＨｚ（最大８逓倍１００ＭＨｚ動作）、サブ３２．７６８ＫＨｚ搭載。
●大容量メモリ内蔵　ＦＬＡＳＨ　１．５ＭＢｙｔｅ、ＲＡＭ　１２８ＫＢｙｔｅ
●コンパクト７３×７３mmサイズにＵＳＢ（ＦＴ２３２ＲＬ）、ＲＳ２３２Ｃ（ＡＤＭ３２０２　２ｃｈ）、ＥＥＰＲＯＭ（２５ＬＣ２５６　３２ＫＢｙｔｅ）ＩＣ搭載。
●動作電圧３．３Ｖ　ＴＹＰＥ　５０ｍＡ（１００ＭＨｚ動作時）
●豊富な周辺機能
Ｉ／Ｏポート：入出力１１７、入力１、内部プルアップ抵抗１１７、オープンドレイン１１７、５Ｖトレラント５３、Ａ／Ｄ変換器：１２ビット、２１ｃｈ、Ｄ／Ａ変換器：１０ビット２ｃｈ、リアルタイムクロック内蔵、シリアル（ＳＣＩ）：１２ｃｈ、ＣＡＮモジュール：３ｃｈ、外部バス拡張（１６ビット）、ＤＭＡ、強力なタイマ：ＭＴＵ２（１６ｂｉｔ×６ｃｈ）、ウオッチドグタイマ、コンペアマッチタイマ（１６ｂｉｔ×２ｃｈ）、温度センサ等。
●デバッカＥ１によるデバック用コネクタ搭載。ＦＩＮＥ接続。

基板大きさ



ｃ：Ｅ１デバッカ



概要

E1 エミュレータは、ルネサス主要マイコンに対応したオンチップデバッギングエミュレータです。基本的なデバッグ機能を有した低価格の購入しやすい開発ツールで、フラッシュプログラマとしても使用可能です。

C 言語ソースデバックが可能で、1 行実行、ブレークポイント設定、変数、レジスタ、メモリ参照等々、従来であれば高価な ICE しか出来なかった機能が、安価に実現されています。また、使い方もＨＥＷ（統合開発環境）の E8a と同じで、経験があれば半日で、無くても 1 日で必要な操作を会得することが出来ると思います。

マイコンとの通信として、シリアル接続方式（FINE)と JTAG 接続方式の２種類に対応しています。使用可能なデバッグインタフェースは、ご使用になるマイコンにより異なります。

また、基本デバッグ機能に加え、ホットプラグイン機能（動作中のユーザシステムに後から E1 エミュレータを接続して、プログラムの動作確認を行うことが可能）を搭載しているため、プログラムのデバッグ･性能評価に大きく貢献できます。

対応MPU

- V850 ファミリ
- RX ファミリ
- RL78 ファミリ
- R8C ファミリ
- 78K ファミリ



Ｅ１を購入するとＣＤが添付されていて、ドライバーのインストールとセルフチェックを行った後に、ネットから開発環境ＨＥＷとＣコンパイラのダウンロードを行います。

E1/E20 USBドライバのインストール

E1またはE20エミュレータをホストマシンに接続する前に、E1/E20用のUSB ドライバをインストールします。このフォルダにあるE1USBDRIVER.exe をクリックしてください。

E1/E20のセルフチェック

ご購入後、製品が正しく動作することを、このフォルダにあるE1/E20自己診断プログラムE1E20SCP.exeを実行して確認してください。
また、ご使用中に起動しない等の問題が発生した場合にも、本プログラムを使用して自己診断を行ってください。

セルフチェックプログラムの実施手順については、こちらを参照ください。故障による、修理、交換については、ルネサス販社または特約店にお問い合わせください。

対応ソフトウェアのインストール

本製品を使用するためには、ご使用になるマイコンに合わせたソフトウェアをインストールする必要があります。

対応マイコン	コンパイラ(必要)	デバッガ(必要)	フラッシュプログラマ (任意)
RXファミリ	RXファミリ用C/C++コンパイラパッケージ 無償評価版 [ダウンロードページへ] 無償評価版のダウンロードページへジャンプします。製品版をお持ちの方は製品版を使用してください。	RXファミリ用エミュレータソフトウェア インストールプログラムを格納したフォルダがオープンしますので、そのフォルダにあるHewInstMan.exeを実行してください。インストールが始まります。	フラッシュ開発ツールキット 無償評価版 [ダウンロードページへ] 無償評価版のダウンロードページへジャンプします。製品版をお持ちの方は製品版を使用してください。

ｃ：無償版ＨＥＷ、ＲＸ用Ｃコンパイラのダウンロード

　プログラムの開発はルネサスエレクトロニクス社の統合開発環境ＨＥＷでＣ言語を用い動作させることができます。ＣＤ添付のサンプルプログラムはこの環境下で作成されています。無償版をダウンロードして使用します。

　ネット検索で→「ＲＸファミリコンパイラダウンロード」などで以下の画面を表示。

ホーム ＞ 製品 ＞ 開発環境 ＞ コーディングツール ＞ コンパイラ/アセンブラ

RXファミリ用C/C++コンパイラパッケージ

製品

開発環境

コーディングツール

コンパイラ/アセンブラ

RXファミリ用C/C++コンパイラパッケージ

SuperHファミリ用C/C++コンパイラパッケージ

統合開発プラットフォーム CubeSuite+

V850用ソフトウェアパッケージ [SP850]

V850用コンパイラパッケージ [CA850]

M32Rファミリ用C/C++コンパイラパッケージ [M3T-CC32R]

R8C, M16Cファミリ用C/C++コンパイラパッケージ

78K0R用ソフトウェアパッケージ [SP78K0R]

78K0R用コンパイラパッケージ [CC78K0R]

78K0R用アセンブラパッケージ [RA78K0R]

78K0用ソフトウェアパッケージ [SP78K0]

78K0用コンパイラパッケージ [CC78K0]

78K0用アセンブラパッケージ [RA78K0]

78K0S用ソフトウェアパッケージ [SP78K0S]

概要 | ドキュメント | ダウンロード | 設計情報/サポート | 関連情報

検索

5件のうち1-5件を表示しています。　表示件数 10

分類	ソフトウェア名	登録日	説明	備考
Device File Updater	RXファミリ用 Device File Updater V.1.04	Oct.05.12		
統合開発環境	統合開発環境 High-performance Embedded Workshop V.4.09.01 フルアップデート	Jun.20.12	コンパイラ、デバッガに付属しているHigh-performance Embedded Workshop のアップデータです。	
統合開発環境	統合開発環境 High-performance Embedded Workshop V.4.09.01 差分アップデート(V.4.09.00から)	Jun.20.12	コンパイラ、デバッガに付属しているHigh-performance Embedded Workshop のアップデータです。V.4.09.00からのみアップデートできます。	
RXコンパイラパッケージ	RXファミリ用C/C++コンパイラパッケージ V.1.02 Release 01 アップデート	Mar.21.12		
RXコンパイラパッケージ	【無償評価版】RXファミリ用C/C++コンパイラパッケージ V.1.02 Release 01	Mar.21.12	High-Performance Embedded WorkshopおよびシミュレータデバッガHを同梱。	

ＨＥＷ＋Ｃコンパイラ同梱をダウンロードします。

　無償版は６０日経過後、リンクサイズが１２８ＫＢと制限されます。
統合開発環境ＨＥＷとＣコンパイラがインストゥールされます。

　ＣコンパイラにＧＣＣ、リンク容量制限無しを希望される場合、インドのＫＰＩＴ社からＨＥＷ環境で動作できるＣコンパイラを無償でダウンロードできます。日本語画面があります。

KPIT Cummins

KPIT GNU TOOLS & SUPPORT

Your Tools Partner

RSS | Twitter | Feedback | News | Contact Us

PLATINUM PARTNER RENESAS

ホーム | 登録 | フリーサポート | 無償ダウンロード | 文書資料 | サイトマップ | 英語の質問 | 検索

日本語 English

KPIT GNUツールとサポート・ウェブサイトへようこそ!

KPIT GNUツールは、ルネサス・マイクロ用のユーザーフレンドリーなGNUツールチェーンであり、当社は、これらについてのカスタマーサポートを無料で世界中に提供しています。
私たちは高品質の製品だけを提供することで満たされなくて、弊社製品のために世界中で無料でルネサスHEW IDE, KPIT Eclipse IDE,との統合と、チュートリアルと、FAQと、専門のテクニカルサポートも提供しています。

ダウンロードとサポートを無制限のアクセスを得るため、今すぐご 登録ください！それは簡単ですよ。
1. 当社のウェブサイトに登録する。
2.私たちはメールであなたのユーザ名とパスワードを送ります。
3. ログインして選択したツールのインストーラをダウンロードする。
4. インストーラを起動する。
5. ルネサスHEW IDE, KPIT Eclipse IDEかMakefiles を使って、アプリケーションを開発する。

ご質問があれば、このウェブサイトから私たちの経験豊富なサポートチームに遠慮せずにご連絡してください。あるいは、あなたは私たちの討議フォーラムでKPIT GNUツールの他のユーザーと話をすることもできます。

KPIT Cumminsインフォシステムズ株式会社は、国際的なソフトウェアソリューションプロバイダーであり、ルネサス・デバイスとソフトウェアに関してコンサルタント業のサービスも提供しています。私たちは、どのようにあなたをサポートすることができるかについて、あなたのインプットを待っています。さらに情報が必要な場合は、または、私たちにご連絡したく時に、ここでクリックしてください。

最新のKPIT GNUツールダウンロード

ダウンロード 最新のKPIT GNUツールチェーンのダウンロード (for Windows)

リリース・ノート | 他のシステムとターゲット

支援された目標

RX | RL78 | V850 | SH | H8 | M16C | M32C

RX600 RX200

ダウンロード数： 30,314 17 Mar 2013 とは KPIT GNU ツール: 専用クライアントとオープンソース開発ツール

ユーザーログイン

ユーザー名:

パスワード:

Go 登録 | パスワードを忘れた

最新アップデート

- KPIT GNURL78 v13.01 のメンテナンスパック1 - リリース済み 28 February 2013
- KPIT GNU Tools GNURL78 v13.01 – リリース済み 13 February 2013
- KPIT GNU Tools GNURX v12.03 – リリース済み 23 November 2012
- KPIT GNU ツール v12.02 - リリース済み 27 June 2012

よりたくさんの

何がNEWです

KPIT Tools チームと働きたいですか？？キャリア/サービス依頼のためにClick Here.

ＫＰＩＴ社はルネサスのプラチナパートナー会社です。

統合開発環境・コンパイラ・コード生成評価支援	ページトップに戻る
IARシステムズ株式会社	IAR Embedded Workbench(EW)
株式会社アイ・エル・シー	組込み機器向けGUI開発環境 GENWARE®4(ジェンウェア フォー) 組込み機器向けGUI開発環境 GENWARE®3(ジェンウェア スリー) 組込みソフトウェア開発支援パッケージ Real-Series(リアル・シリーズ)
ガイオ・テクノロジー株式会社	関数内変数アクセス情報出力ツール「FuncVarGrid」 NEW! 自動車機能安全規格ISO26262 ツール認証取得 C/C++組み込み向け単体テストツール「カバレッジマスター winAMS」 クロス統合開発環境「XASS-Vシリーズ / フレームワーク」 組み込み用プログラムチャート自動生成ツール「CasePlayer2」
KPIT	GNU Cコンパイラ(英語版)
テクマトリックス株式会社	C/C++対応静的解析・動的解析ツール C++test
株式会社東陽テクニカ	C/QA・C++ C/C++言語静的解析ツール
日本アイ・ビー・エム株式会社	Rational DOORS Rational Team Concert Rational Quality Manager
株式会社リンクス	IEC61131-3準拠のソフトPLC・ソフトMotion・HMI「CODESYS」 NEW!
協調検証ツール	ページトップに戻る
株式会社ガイア・システム・ソリューション	GAIA製高速プロセッサモデル・ライブラリ -RX600コア SystemCモデル- NEW!
	組み込みシステム統合検証環境「No.1 システムシミュレータ」

上記、純正Ｃ、ＫＰＩＴ　Ｃ（ＧＣＣ）コンパイラはいずれも高速で、ＨＥＷ，Ｅ１の優れた操作性と相まって先進的な開発環境を構築できます。

ｄ：開発セット添付ＣＤコピー、デバイスドライバＤ２ＸＸのインストゥール

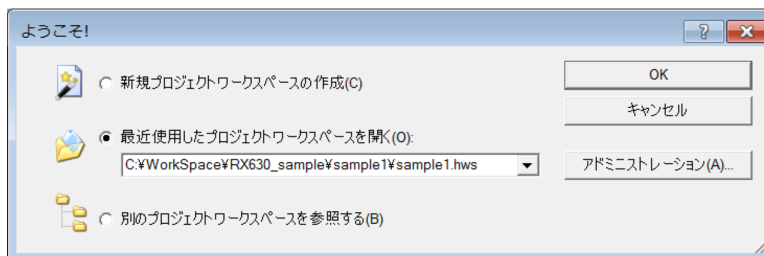
省略

１－２　　動作、デバック

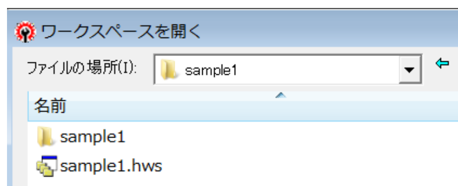
ａ：ＨＥＷ起動、コンパイル、書き込み、動作



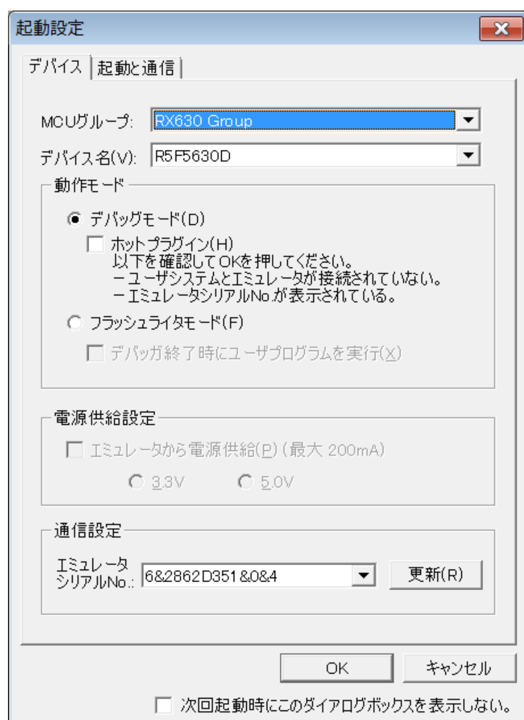
ＨＥＷを起動します。ここでは例としてＲＸ６３０￥ｓａｍｐｌｅ１を動作させます。初めてのときは「別のプロジェクトワークスペースを参照する」を選択し



拡張子ｈｗｓファイルをダブルクリック。以降、同じｓａｍｐｌｅ１でしたら「最近使用した、、」でＯＫです。



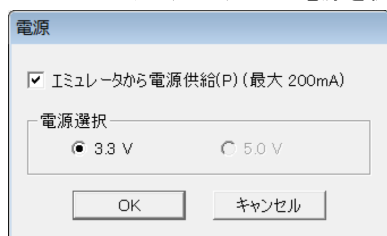
起動設定が表示されます。「ＯＫ」をクリック。



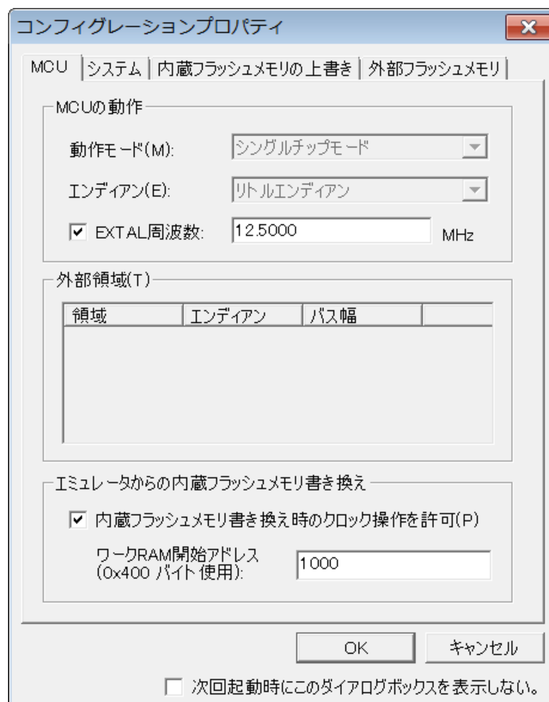
サンプルプログラム実行にあたって、電源はＥ１から供給されますので、他に用意する必要はありません。「ＯＫ」をクリック。



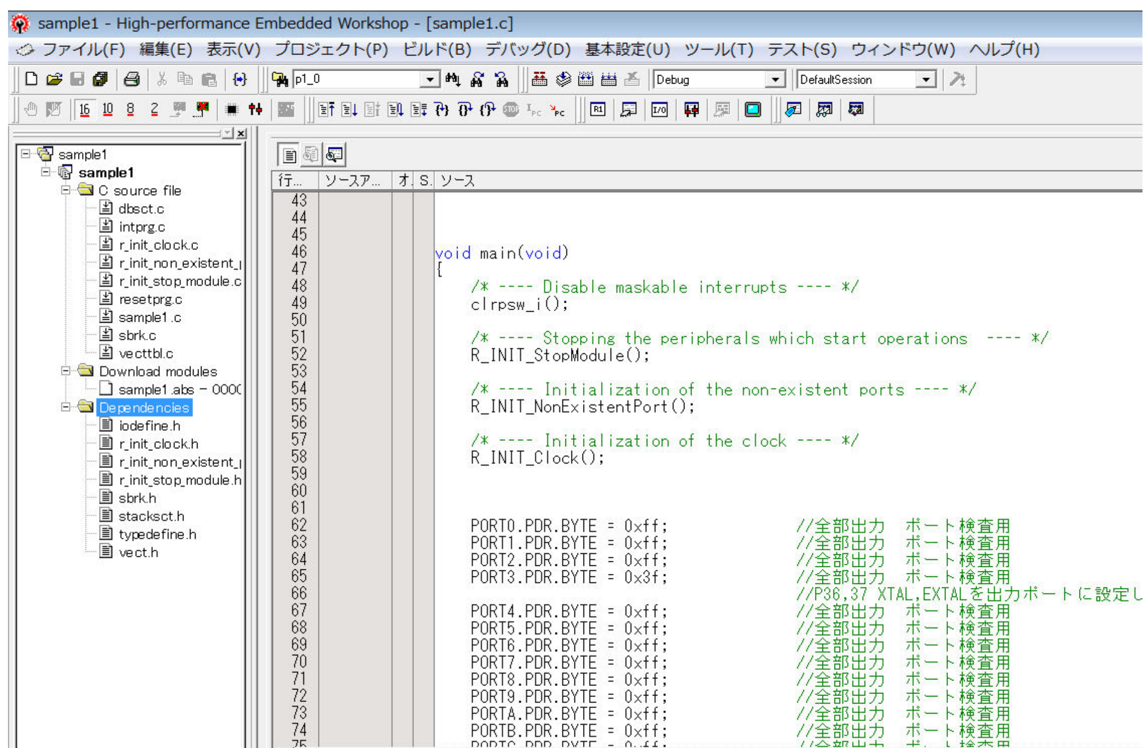
エミュレータ（Ｅ１）から電源を供給するにチェック、３．３Ｖを選択し「ＯＫ」をクリック。



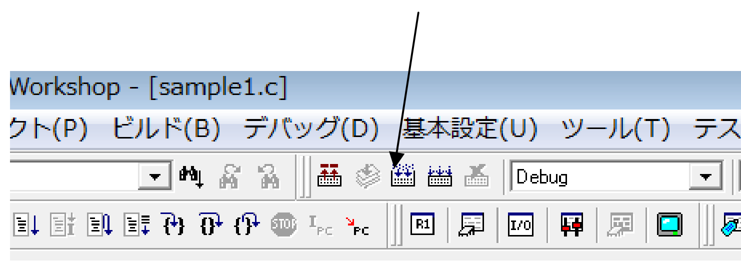
コンフィグレーションプロパティが表示されます。ＥＸＴＡＬ周波数にチェック、１２．５ＭＨｚ、「内蔵フラッシュメモリ､､」にチェックが入っている必要があります。確認後、「ＯＫ」。



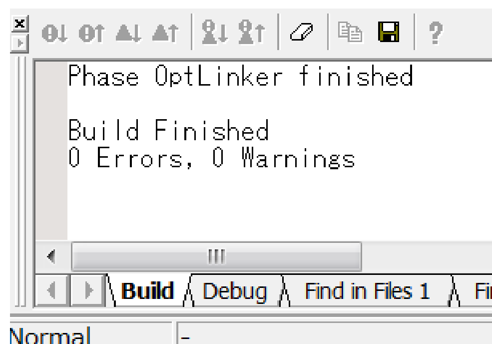
ｓａｍｐｌｅ１のエディタ画面等が表示されます。



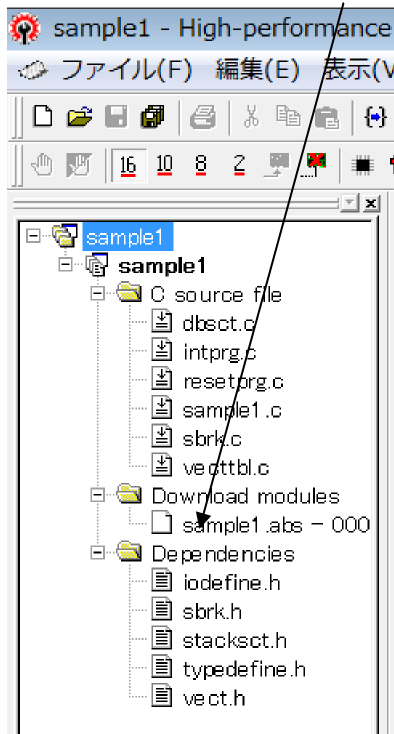
プログラムのコンパイルは、以下の「ビルド」ボタンで行います。ボタンはマウスを乗せると意味が表示されます。



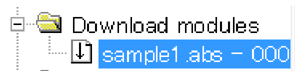
ｓａｍｐｌｅ１は出荷時にすでにコンパイラされていますので、０　Errors、０　Warnings と表示されます。新たにプログラムを製作してコンパイルしたときに Errors が０で無い場合、プログラムに文法上の問題等ありますので、エディタでソースファイルを修正し、Errors　０にする必要があります。



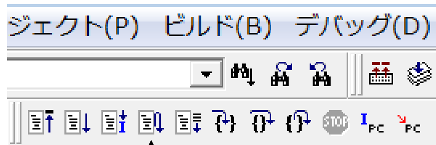
ｓａｎｐｌｅ１．ａｂｓ　　　をダブルクリックします。



正常にダウンロードされると空白だった部分に↓マークが入ります。

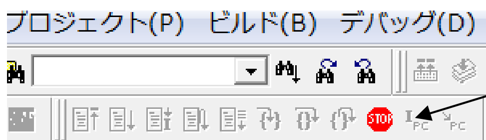


実行させてみます。



「リセット後実行」をクリック

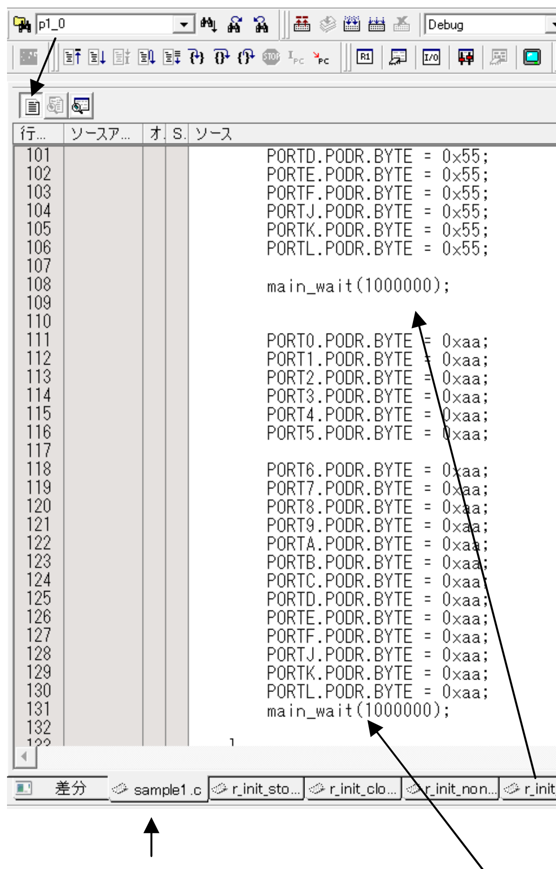
ＣＰＵボードのＬＥＤが点滅しているのが見えると思います。停止はＳＴＯＰをクリックします。



以上がプログラム開発に必要な「コンパイル」「書き込み」「実行」です。

例としてｓａｍｐｌｅ１を書き換えて、コンパイル、書き込み、動作の変化の確認、を行います。

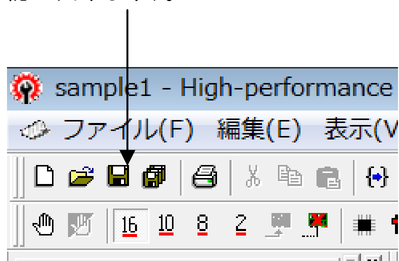
Ｃソースファイルが選択されていること



ｓａｍｐｌｅ１．ｃが選択されていること。

ｌｗａｉｔ(1000000)の中の数値を 1 桁増やし 10000000（0 を７個）にしてみます（２箇所)。プログラムをセーブします。

ファイルの保存はここをクリックします。エディタで書き込み、保存すると色が変わります（未保存の確認が出来ます)。



次ぎに、「コンパイル」し、エラーが無い場合、以下が表示されます。はい（Ｙ）をクリックでプログラムがダウンロードされます。



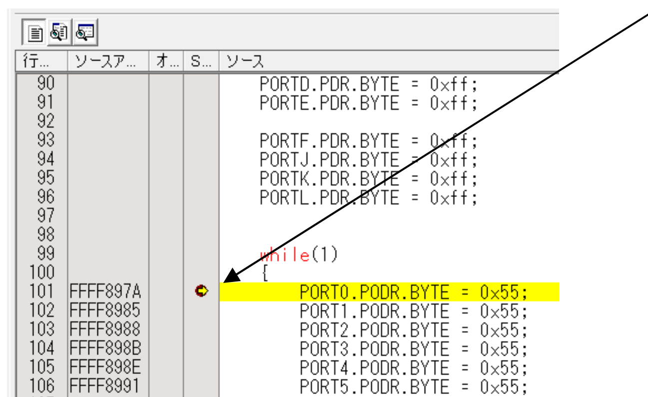
「リセット後、実行」をクリックします。ＬＥＤの点滅周期が遅くなったのがおわかりいただけると思います。

以上のように、プログラム開発は「エディタ（プログラム作成)」→「セーブ」→「コンパイル」→「エラーが無いことを確認」→「書き込み」→結果によって頭の「エディタ」に戻る繰り返しになります。

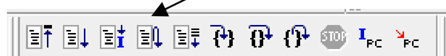
エディタは使い慣れたものでも使用可能で、その場合、ＨＥＷのエディタは使えなくなります。

ｂ：ブレークポイント設定、レジスタ、変数参照概要

ブレークポイントはＳ/Ｗブレークポイントの部分をダブルクリックすることにより、設定、解除できます。



設定後、「リセット後実行」をクリックするとプログラム動作はブレークポイントで停止し、カーソルが黄色になります。

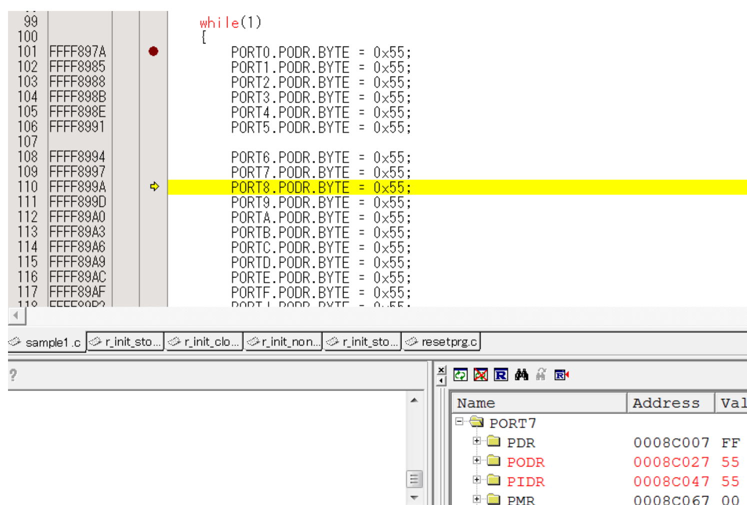


ステップインで１づつ実行。



ここで例えば、ＰＯＲＴ７（ＬＥＤ４が接続されているポート）の内容を見てみます。Ｉ／Ｏをクリック。

ステップインを繰り返しクリック。ＰＯＲＴ８に命令したところで、Ｉ／Ｏ窓のＰＯＤＲ（ポート出力レジスタ）に０ｘ５５が書き込まれ、ＬＥＤ　Ｄ４が点灯することが確認出来ます。黄色いカーソルがある行はまだその命令は実行されていませんので、ご注意願います。



さらにステップインを繰り返すとｍａｉｎ＿ｗａｉｔ（）関数にジャンプします。

```
50  FFFF8935      void main_wait(long ltime)
51                {
52  FFFF8938          while(ltime != 0)
53                    {
54  FFFF8936              ltime--;
55                    }
56                }
```

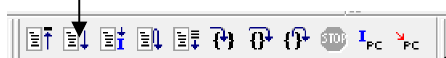
ここでローカル変数ｌｔｉｍｅを確認してみます。ローカルをクリック。



ローカル変数窓が開きます。ステップインを繰り返すたびにｌｔｉｍｅが−１されるのが確認出来ます。

Name	Value	Type
ltime	D'999989 ...	(long)

メインプログラムに戻るためにブレークポイントを設定し、「実行」。この実行は、現在のプログラムカウンタからの実行です。



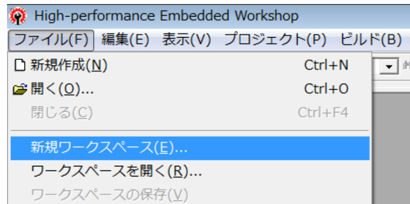
```
121
122  FFFF897D        main_wait(1000000);
123
124
125  FFFF89BE        PORT0.PODR.BYTE = 0xaa;
126  FFFF89C9        PORT1.PODR.BYTE = 0xaa;
127  FFFF89CC        PORT2.PODR.BYTE = 0xaa;
128  FFFF89CF        PORT3.PODR.BYTE = 0xaa;
129  FFFF89D2        PORT4.PODR.BYTE = 0xaa;
130  FFFF89D5        PORT5.PODR.BYTE = 0xaa;
```

ｍａｉｎ＿ｗａｉｔ（）ルーチンを抜けました。

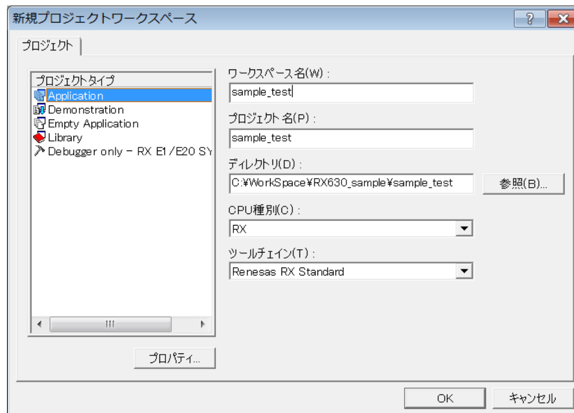
以上がデバックの概要です。ＨＥＷ、Ｅ１の詳しい使い方は「Ｅ１アプリケーションノート」をダウンロードし、ご参照下さい。

ｃ：新しいプログラムを作る

下記の例では WorkSpace￥RX630_sample￥というホルダに sample_test というプログラムを作成する、という前提で説明します。



ＣＰＵ種別はＲＸにします。ツールチェーンは自動的にＲＸ　Ｓｔａｎｄａｒｄになります。



省略

２－３　ｓａｍｐｌｅ３　Ａ／Ｄ変換をＵＳＢ、ＲＳ２３２Ｃ出力

```
/***********************************************************************/
/*                                                                     */
/*  FILE        :sample3.c                                             */
/*  DATE        :Sat, Feb 15, 2014                                     */
/*  DESCRIPTION :Main Program                                          */
/*  CPU TYPE    :RX630                                                 */
/*                                                                     */
/*  This file is generated by Renesas Project Generator (Ver.4.53).    */
/*  NOTE:THIS IS A TYPICAL EXAMPLE.                                    */
/*                                                                     */
/***********************************************************************/
/*
A/D  RS232C_0 出力
2014.2.15

         【  動作  】
                 AD0  P40  CN2  8 番の電圧を AD 変換し USB で出力する
                 AD1  P41  CN2  10 番の電圧を AD 変換し RS232C_0 で出力する
         【  接続  】PC  <->  USB ミニケーブル  または  PC <-> Ｋケーブル  両方でも可。
         【  事前設定  】
                 １．USB ミニケーブル、Ｋケーブルで RX630 ボードとパソコンを接続。
                 ２．HEW で sample3.mot を RX630 ボードに書き込み動作させる。
                 ３．USB はパソコンが FTDI デバイスの認識終了後、テラタームを動作させる。
         【  注意  】
         USB 接続の場合、スパイク的な A/D 値の変動が起こる現象あり。パソコン側のノイズ？
         精密測定の場合要注意。
*/

#include <machine.h>
#include "iodefine.h"
#include "r_init_clock.h"
#include "r_init_non_existent_port.h"
#include "r_init_stop_module.h"
#include "sio_RX630.c"


#define CR 0x0d
#define LF 0x0a

unsigned char cf;
unsigned short eep_data;
unsigned long ldata1;

unsigned short sdata0,sdata1;
```

```
unsigned char c1,c2;
```

省略

```
void main(void)
{
```

省略（既に説明済みのため）

```
//AD 初期設定

①        SYSTEM.PRCR.WORD = 0xA502;              //レジスタライトプロテクション解除  PRC1
          SYSTEM.MSTPCRA.BIT.MSTPA17 = 0;         //AD モジュールストップ解除
          SYSTEM.PRCR.WORD = 0xA500;              //レジスタライトプロテクション有効

          PORT4.PDR.BIT.B0 = 0;                   //入力
          PORT4.PDR.BIT.B1 = 0;                   //入力

          PORT4.PMR.BIT.B0 = 1;                   //P40 周辺機器
          PORT4.PMR.BIT.B1 = 1;                   //P41 周辺機器

          MPC.PWPR.BIT.B0WI = 0;                  //PFS レジスタ書き込み１
          MPC.PWPR.BIT.PFSWE = 1;                 //PFS レジスタ書き込み２許可

          MPC.P40PFS.BIT.ASEL = 1;                //AD0=P40 端子割り振り
          MPC.P41PFS.BIT.ASEL = 1;                //AD1=P41 端子割り振り

          MPC.PWPR.BIT.PFSWE = 0;                 //PFS レジスタ書き込み１禁止
          MPC.PWPR.BIT.B0WI = 1;                  //PFS レジスタ書き込み２禁止

          S12AD.ADEXICR.WORD = 0x0000;            //温度、基準電圧変換 ADx の変換時は 0

          char_out0('T');
          char_out0('E');
          char_out0('S');
          char_out0('T');
          char_out0(' ');
          char_out0('A');
          char_out0('D');

    while(1)
    {

                    PORT7.PODR.BIT.B0 = 1;
```

```
②                            S12AD.ADANS0.BIT.ANS0 = 3;           //AD0,1 選択
                             S12AD.ADCSR.BIT.ADST = 1;            //AD 開始

                             while(S12AD.ADCSR.BIT.ADST == 1)//変換終了待ち
                                     ;
//USB 出力
                             char_out6('A');
                             char_out6('D');
                             char_out6('0');
                             char_out6('=');
③                            sdata0 = S12AD.ADDR0;
                             ascout6_16(sdata0);
                             char_out6(CR);
                             char_out6(LF);

//RS232C_0 出力
                             sdata0 = S12AD.ADDR1;
                             char_out0('A');
                             char_out0('D');
                             char_out0('1');
                             char_out0('=');
                             ascout0_16(sdata0);
                             char_out0(CR);
                             char_out0(LF);


                    PORT7.PODR.BIT.B0 = 0;
                    main_wait(3000000);

    }

}
```

【　解説　】

```
①        SYSTEM.PRCR.WORD = 0xA502;            //レジスタライトプロテクション解除　PRC1
          SYSTEM.MSTPCRA.BIT.MSTPA17 = 0;       //AD モジュールストップ解除
          SYSTEM.PRCR.WORD = 0xA500;            //レジスタライトプロテクション有効
```

始めにＡ／Ｄを使える状態にするためにモジュールストップを解除します。入力指定、ＰＭＲで周辺機器指定します。

```
          PORT4.PDR.BIT.B0 = 0;                 //入力
          PORT4.PDR.BIT.B1 = 0;                 //入力
```

```
PORT4.PMR.BIT.B0 = 1;               //P40 周辺機器
PORT4.PMR.BIT.B1 = 1;               //P41 周辺機器
```

ＰＦＳレジスタでポートの端子をＡＳＥＬ（アナログ端子として使用する）を選択します。

```
MPC.PWPR.BIT.B0WI = 0;              //PFS レジスタ書き込み１
MPC.PWPR.BIT.PFSWE = 1;             //PFS レジスタ書き込み２許可

MPC.P40PFS.BIT.ASEL = 1;            //AD0=P40 端子割り振り
MPC.P41PFS.BIT.ASEL = 1;            //AD1=P41 端子割り振り

MPC.PWPR.BIT.PFSWE = 0;             //PFS レジスタ書き込み１禁止
MPC.PWPR.BIT.B0WI = 1;              //PFS レジスタ書き込み２禁止

S12AD.ADEXICR.WORD = 0x0000;        //温度、基準電圧変換 ADx の変換時は０
```

```
②                  S12AD.ADANS0.BIT.ANS0 = 3;          //AD0,1 選択
                    S12AD.ADCSR.BIT.ADST = 1;           //AD 開始

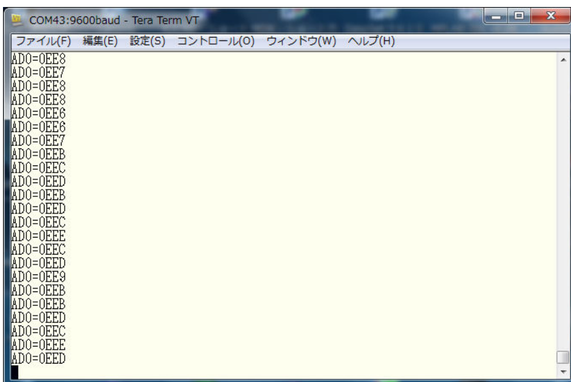
                    while(S12AD.ADCSR.BIT.ADST == 1)//変換終了待ち
                             ;
```

ＡＤ０，１入力を選択し、変換開始、変換終了待ちです。

```
//USB 出力
                    char_out6('A');
                    char_out6('D');
                    char_out6('0');
                    char_out6('=');
③                  sdata0 = S12AD.ADDR0;
                    ascout6_16(sdata0);
                    char_out6(CR);
                    char_out6(LF);
```

ＡＤ０＝　と文字を出力し、さらにＡ／ＤデータをＡＳＣＩＩ変換してＵＳＢに出力しています。

ＡＤ０、１端子の電圧を変えたり、＋３．３Ｖ、０Ｖに接続すると値が変わるのが確認出来ます。



２－６　三角、対数、平方根関数を使う

```
/**********************************************************************/
/*                                                                    */
/*  FILE        :sample6.c                                            */
/*  DATE        :Tue, Feb 18, 2014                                    */
/*  DESCRIPTION :Main Program                                         */
/*  CPU TYPE    :RX630                                                */
/*                                                                    */
/*  This file is generated by Renesas Project Generator (Ver.4.53).   */
/*  NOTE:THIS IS A TYPICAL EXAMPLE.                                   */
/*                                                                    */
/**********************************************************************/

/*
三角関数

            【　動作　】log、sin、ルートの演算を行い、double に浮動小数点格納されます。数値と演算時間を確認します。

            【　接続　】特になし

            【　事前設定　】特になし

            【　注意　】特になし

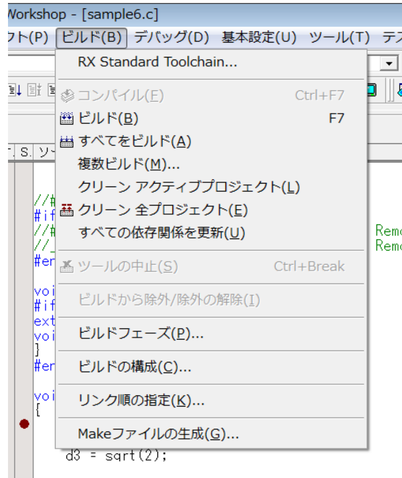

*/
```

省略

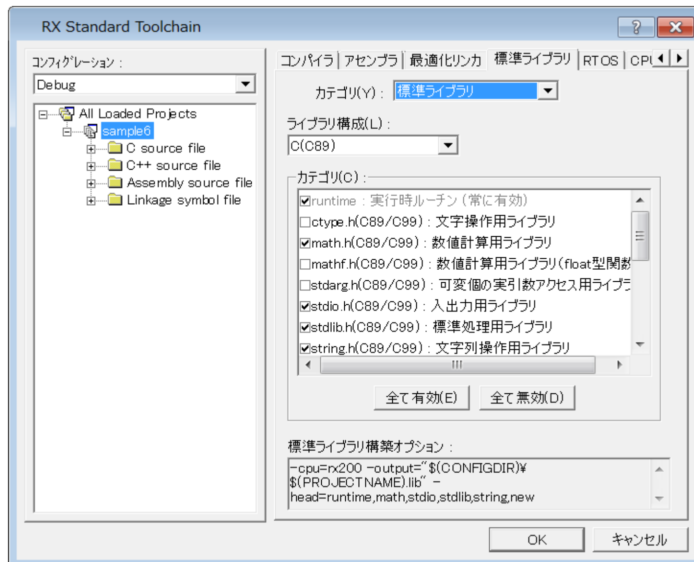
【　解説　】

①#include <math.h>

三角関数や、対数、平方根を使うためにはｍａｔｈ．ｈをインクルードする必要があります。加えて、ビルド（Ｂ）→　ＲＸ　Ｓｔａｎｄａｒｄ　Ｒｏｏｌｃｈａｉｎ､､をクリック。



カテゴリを「標準ライブラリ」にして、ｍａｔｈ．ｈにチェックを入れる必要があります。本サンプルは既に設定済みです。



②double d1,d2,d3;

short s1,s2,s3;

結果を入れるｄｏｕｂｌｅ（６４ビット）３つ、キャストするｓｈｏｒｔ（１６ビット）３つをここで定義しています。ｄｏｕｂｌｅは倍精度を指定していますので、３２ビットではなく、６４ビットになります。（設定はｃ：新しいプログラムを作るを参照）

③#define PI 3.14159265

三角関数計算で角度を入力して数値を出すために使います。

④

```
PORT7.PODR.BIT.B0 = 1;          //時間マーカーON
d1 = log10(10000);
PORT7.PODR.BIT.B0 = 0;          //時間マーカーON

PORT7.PODR.BIT.B0 = 1;          //時間マーカーON
d2 = sin((PI/180)*45);
PORT7.PODR.BIT.B0 = 0;          //時間マーカーON

PORT7.PODR.BIT.B0 = 1;          //時間マーカーON
d3 = sqrt(2);
PORT7.PODR.BIT.B0 = 0;          //時間マーカーON
```

頭から

ｄ１＝ｌｏｇ１０（１００００）、答えは４になるはずです。

ｄ２＝ｓｉｎ（(ＰＩ／１８０＊４５）→ｓｉｎ（４５°）という意味です。答えは０．７０７１０６、、になるはずです。

ｄ３＝ｓｑｒｔ（２）→平方根の２です。答えは１．４１４２１３５６になるはずです。

一度「リセット後実行」させ、その後、停止させると右下のワッチウインドウに演算結果が表示されます。

Name	Value	Address	Type	Scope
d1	4	{ 00001008 }	(double)	[Global]
d2	0.707106780551956	{ 00001010 }	(double)	[Global]
d3	1.4142135623731	{ 00001018 }	(double)	[Global]
s1	H'0004	{ 00001000 }	(short)	[Global]
s2	H'0000	{ 00001002 }	(short)	[Global]
s3	H'0001	{ 00001004 }	(short)	[Global]

Watch1 Watch2 Watch3 Watch4

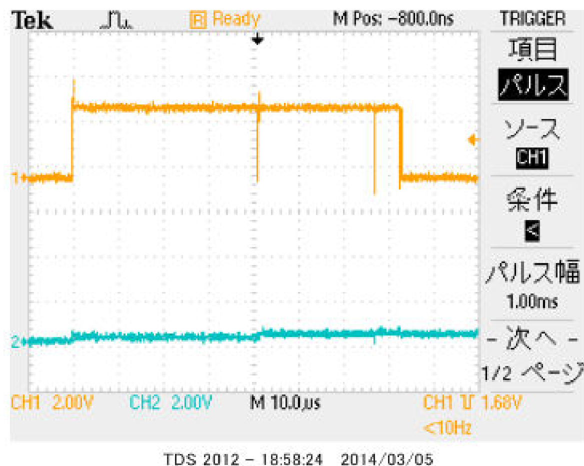
上からｄ１、ｄ２、ｄ３の１０進数倍精度データです。答えは合っていますね。

⑤

```
s1 = d1;
s2 = d2;
s3 = d3;
```

例えば演算結果をＤＡコンバータに出力する場合、浮動小数点のままでは設定できません。⑤は浮動小数点データを整数の１６ビットに設定（キャスト）しています。１０００番地からｓ１，ｓ２，ｓ３です。結果を見ると小数点部分が欠落して設定されていることが分かると思います。(上図　下　ｓ１、ｓ２、ｓ３）

小数点以下何桁まで使用したいか、ということで、ｄｏｕｂｌｅデータを加工してからｓｈｏｒｔに移せば最大の精度、有効数値が得られます。



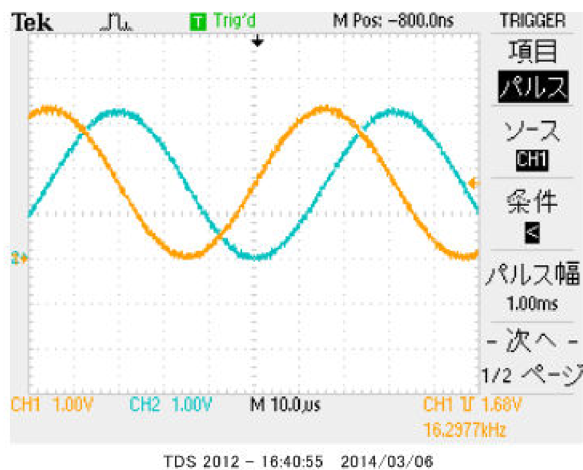
Ｐ７０の端子をオシロで観測すると、図のような波形が得られます。ｌｏｇの演算が約４０μｓｅｃ、ｓｉｎが２６μｓｅｃ、平方根が５μｓｅｃくらいでしょうか。

３２ＭＨｚ動作のＲＬ７８１０４マイコンでｌｏｇ１０（１００００）が約２２０μｓｅｃ、ｓｉｎ（４５°）が１３０μｓｅｃ、√２が１００μｓｅｃ程度かかるので、演算に関してＲＸはクロック比以上の劇的な速さが得られるのが分かります。

※ＲＸは倍精度演算の速度です。ＲＬ７８は単精度演算の速度です。
※ＲＬ７８の演算、ポート制御は従来のＨ８、Ｒ８Ｃマイコン等に比べて何倍も高速です。速度比較詳細は無償ダウンロード出来る弊社「ＲＬ７８１０４の開発セットマニュアル抜粋」でご確認下さい。

２－７　Ｄ／Ａコンバータ　sin、cos 値を出力してみる

【　動作　】



TDS 2012 - 16:40:55　2014/03/06

ＲＸ６３０は分解能１０ビットＤ／Ａ出力を２ｃｈ持っています。そこにｓｉｎ（），ｃｏｓ（）の０～３６０°を演算し、Ｄ／Ａ出力し、電圧をみてみます。いわゆる、正弦波発振器と同じ出力が得られます。

【　プログラム　】

```
/***********************************************************************/
/*                                                                     */
/*  FILE        :sample7.c                                             */
/*  DATE        :Tue, Feb 18, 2014                                     */
/*  DESCRIPTION :Main Program                                          */
/*  CPU TYPE    :RX630                                                 */
/*                                                                     */
/*  This file is generated by Renesas Project Generator (Ver.4.53).    */
/*  NOTE:THIS IS A TYPICAL EXAMPLE.                                    */
/*                                                                     */
/***********************************************************************/
/*

ｓｉｎ、ｃｏｓ値をＤ／Ａコンバータで出力する

        【　動作　】

                0～360°の sin、cos 値を演算し、D／A コンバータで出力する。
                クリスタル精度の極めて安定した正弦波出力が得られます。

        【　接続　】特になし

        【　事前設定　】P03,P05 から波形が出力されるので、オシロスコープで観測できます。

        【　注意　】特になし
```

```
*/

省略

double d1,d2;
short kakudo;

unsigned short sin_data[370],cos_data[370];


#define PI 3.14159265

void main(void)
{
 省略

//D/A コンバータ

①	SYSTEM.PRCR.WORD = 0xA502;		//レジスタライトプロテクション解除　PRC1
	SYSTEM.MSTPCRA.BIT.MSTPA19 = 0;	//DA コンバータモジュールストップ解除
	SYSTEM.PRCR.WORD = 0xA500;		//レジスタライトプロテクション有効

//ポート設定

	PORT0.PDR.BIT.B3 = 1;			//P03 出力
	PORT0.PDR.BIT.B5 = 1;			//P05 出力

	PORT0.PMR.BIT.B3 = 1;			//P03 周辺機器
	PORT0.PMR.BIT.B5 = 1;			//P05 周辺機器

	MPC.PWPR.BIT.B0WI = 0;			//PFS レジスタ書き込み１
	MPC.PWPR.BIT.PFSWE = 1;			//PFS レジスタ書き込み２許可

	MPC.P03PFS.BIT.ASEL = 1;		//P03 DA0
	MPC.P05PFS.BIT.ASEL = 1;		//P05 DA1

	MPC.PWPR.BIT.PFSWE = 0;			//PFS レジスタ書き込み１禁止
	MPC.PWPR.BIT.B0WI = 1;			//PFS レジスタ書き込み２禁止

	DA.DACR.BYTE = 0xdf;			//D/A コンバータ出力許可


//PORT 初期化

	PORT7.PDR.BYTE = 0x01;				//出力　ＬＥＤ用
```

```
//演算して結果をメモリにセーブ

②                  for(kakudo = 0;kakudo < 360 ;kakudo++)
                   {
                           d2 = sin((PI/180)*kakudo);          //1から－1まで変動
                           d2 +=1;
                                             //オフセット＋１→  0～2の変化になる
                           d2 *= 511.5;      //2を最大電圧3.3Vにする。
                           sin_data[kakudo] = d2;              //sin信号をDAOUT P03


                           d1 = cos((PI/180)*kakudo);          //１から－１まで変動
                           d1 += 1;                            //オフセット＋１
                           d1 *= 511.5;                        //2を3.3Vにする
                           cos_data[kakudo] = d1;              //cos信号をDAOUT P05
                   }

//演算結果をD/Aに出力

 ③  while (1U)
    {
                   PORT7.PODR.BIT.B0 = 1;                      //時間測定マーカーＯＮ

                   for(kakudo = 0;kakudo < 360 ;kakudo++)
                   {
                           DA.DADR0 = sin_data[kakudo];        //sin信号をDAOUT P03

                           DA.DADR1 = cos_data[kakudo];        //cos信号をDAOUT P05
                   }

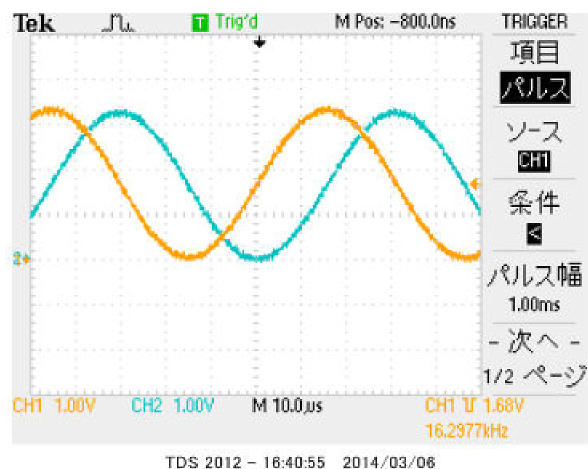
                   PORT7.PODR.BIT.B0 = 0;                      //時間測定マーカーＯＦＦ
    }

}
```

【　解説　】

省略

演算とＤＡ出力を分離したことにより、



１６．２９７７ｋＨｚという正弦（ｓｉｎ）波、余弦（ｃｏｓ）波が得られました。この周波数はクリスタルの精度で、極めて安定しています。周波数を低くするには１データ出力毎にウエイトを入れることで可能です。人間の可聴帯域はほぼカバーすることが出来ます。ＣＲ発振器が苦手な超低周波信号も高精度、高安定で作成できます。

WindowsXPⓇ、WindowsVistⓇ、Windows7Ⓡはマイクロソフト社の登録商標です。
フォースⓇライタは弊社の登録商標です。

１．本文章に記載された内容は弊社有限会社ビーリバーエレクトロニクスの調査結果です。
２．本文章に記載された情報の内容、使用結果に対して弊社はいかなる責任も負いません。
３．本文章に記載された情報に誤記等問題がありましたらご一報いただけますと幸いです。
４．本文章は許可なく転載、複製することを堅くお断りいたします。

お問い合わせ先：

〒３５０－１２１３　埼玉県日高市高萩１１４１－１
TEL　０４２（９８５）６９８２
FAX　０４２（９８５）６７２０
Homepage：http//beriver.co.jp
e-mail：info@beriver.co.jp